Bond Graph Representation of Nuclear Reactor Point Kinetics and Nearly Incompressible Thermal Hydraulics

Eugeny Sosnovsky*, Benoit Forget*

Computational Reactor Physics Group, Department of Nuclear Science and Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, United States

Abstract

This work presents a simplified 1D model for a pressurized water reactor core, suitable for very rapid transients like control rod ejection. The model is represented using the bond graph formalism, a technique for modeling engineering systems as combinations of connected elements. Bond graphs are a flexible way of presenting coupled physics problems by automating the computer science aspects of modeling and letting the modelers focus on the physics; they were introduced in earlier work.

To help leverage the flexibility of bond graph representations of physical systems, a new bond graph processing code, BGSolver, is introduced. BG Solver has been developed by the authors over the past several years, and is now released as open source software.

A rapid rod ejection benchmark is solved with both BG Solver and RELAP5-3D; BG Solver obtained full convergence with a 5 millisecond time step, while RELAP5-3D required a 1 millisecond time step, due to the fully coupled time integration that BG Solver employed, compared to an operator splitting-based time integrator of RELAP5-3D. BG Solver’s time integrator demonstrated 3rd-order convergence in time, a very desirable property. A single nonlinear solve was used to obtain the steady state with BG Solver.

Keywords: Reactor multiphysics, Bond graphs, Point kinetics, Incompressible flow

1. Introduction

In nuclear reactor transients, the two most important physics are neutron transport and thermal hydraulics. These physics are coupled: fission creates a significant heat source in a reactor, while temperature and density variations (particularly moderator density) affect reactor reactivity.

Depending on the desired fidelity, transient neutron transport can either be treated as a full spatially distributed problem (spatial neutron kinetics), or as a point kinetics problem, in which only the amplitude of the fission power source in the reactor varies. The assumption of a steady shape is the limiting one in such models. They are rarely used for detailed safety analysis, however, they are much simpler to construct and solve, and so are frequently used for:

1. Operator training and examination: the point kinetics model is sufficiently small to be solved in real time, which allows it to be used together with systems codes in training scenarios.

2. Coarse systems-level thermal hydraulic models for safety transient analysis, particularly the ones done by a single thermal hydraulic code package; such codes typically do not have full flexible spatial kinetics capabilities. One notable example is RELAP5-3D, which incorporates a full point kinetics representation (The RELAP5-3D Code Development Team 2005).

*Corresponding Authors. Address: Massachusetts Institute of Technology, 77 Massachusetts Avenue, Room 24-214, Cambridge, MA 02139, United States. Tel.: +1 617 253 1655 (B. Forget).

Email addresses: esos@mit.edu (Eugeny Sosnovsky), bforget@mit.edu (Benoit Forget)
3. Control system design and optimization.

Whether or not point or spatial kinetics are used, thermal hydraulics and neutron transport coupling is a complicated multiphysics problem. Such problems are normally treated through operator splitting, but recently, there has been a push to use fully coupled simulation instead (Knoll and Keyes 2004; Gaston et al. 2009). One of the approaches used to develop non-physics-specific fully coupled code packages is the bond graph formalism: a set of techniques for representing the problem’s physics in terms of a connected system of graph elements, which is then automatically converted into a state derivative vector and integrated. The bond graph formalism itself, the motivations for its use, as well as its application to spatial neutron kinetics with nonlinear thermal feedback, are described in detail in Sosnovsky and Forget (2013).

One of the earliest applications of bond graphs to reactor analysis was the bond graph representation of a linear point kinetics model with a lumped parameter thermal feedback (Tylee 1981, 1986). At the time, bond graph processing codes could only process fully linear, constant coefficient bond graph models, which lead to a linearized model being used. The code ENPORT (Rosenberg 1985) was used in Tylee (1981). This model has significant inaccuracies for a number of transients, and so, if a bond graph-based model is to be used for any of the above applications today, it is necessary to:

1. Convert the bond graph representation of a linearized point kinetics model with a lumped parameter thermal feedback to a nonlinear point kinetics model with a spatially distributed feedback.

2. Develop a bond graph processing code capable of handling nonlinear bond graph models.

The present paper accomplishes these two objectives. A nonlinear point kinetics model with a 1D nearly incompressible flow feedback is summarized in section 2. A nonlinear bond graph processing code BGSolver, previously used in Sosnovsky and Forget (2013), and recently released as open source software, is presented in section 4. A benchmark problem, solved both using the bond graph representation with BGSolver, and RELAP5-3D, is given in section 5. Section 6 details the paper’s conclusions and future work.

2. Nonlinear point kinetics model with spatially distributed feedback

A nonlinear point kinetics model of a reactor with a simplified spatially distributed feedback consists of two parts: the point kinetics equations system, and the thermal hydraulic equations. These parts are discussed in sections 2.1 and 2.2 respectively.

2.1. Nonlinear point kinetics equations

Assuming a constant spatial shape of the flux, a constant spectrum and constant delayed neutron fractions, the point kinetics of a system with thermal feedback are represented by the following equations:

\[
\frac{d}{dt}N(t) = \left(\frac{\rho(t) - \beta}{\Lambda}\right)N(t) + \sum_{m=1}^{M} \lambda_m C_m(t),
\]

\[
\frac{d}{dt}C_m(t) = \left(\frac{\beta_m}{\Lambda}\right)N(t) - \lambda_m C_m(t) \quad \forall m \in [1, \ldots, M],
\]

in which \(N(t)\) is the number of neutrons in the core at time \(t\), \(\rho(t)\), \(\beta\) and \(\Lambda\) are the reactor reactivity, delayed neutron fraction and prompt neutron generation time, respectively; \(m\) is the precursor family index (1 to \(M\)), and \(\beta_m\), \(\lambda_m\) and \(C_m(t)\) are the precursor family \(m\) delayed neutron fraction, decay constant and number of precursor nuclei in the core, respectively.

Under the above assumptions, \(P(t)\), the power generated in the core, is linearly proportional to the number of neutrons in the core through a constant \(K\). Similarly, the precursor family \(m\) delayed neutron power \(\tilde{C}_m(t)\) (a derived quantity), is also linearly proportional to \(C_m(t)\), through the same constant:

\[
P(t) = KN(t),
\]

\[
\tilde{C}_m(t) = KC_m(t).
\]
Multiplying Eqs. (1) by $K$ yields:

$$\frac{d}{dt}P(t) = \left(\frac{\rho(t) - \beta}{\Lambda}\right) P(t) + \sum_{m=1}^{M} \lambda_m \tilde{C}_m(t), \quad (3a)$$

$$\frac{d}{dt} \tilde{C}_m(t) = \left(\frac{\beta_m}{\Lambda}\right) P(t) - \lambda_m \tilde{C}_m(t) \quad \forall m \in [1, \ldots, M]. \quad (3b)$$

For the purposes of bond graph representation (section 3), it is convenient to rewrite Eq. (3b) in terms of $\tilde{C}_m^*(t)$:

$$\tilde{C}_m^*(t) = \frac{\Lambda}{\beta_m} \tilde{C}_m(t). \quad (4)$$

Substituting Eq. (4) into Eqs. (3) yields:

$$\frac{d}{dt}P(t) = \left(\frac{\rho(t) - \beta}{\Lambda}\right) P(t) + \sum_{m=1}^{M} \lambda_m \frac{\beta_m}{\Lambda} \tilde{C}_m^*(t), \quad (5a)$$

$$\frac{d}{dt} \tilde{C}_m^*(t) = P(t) - \lambda_m \tilde{C}_m^*(t). \quad (5b)$$

Feedback in point kinetics models varies in complexity, but generally consists of the reactor reactivity $\rho(t)$ being dependent on one or more of the following parameters:

- Fuel temperature (fuel Doppler feedback).
- Moderator temperature.
- Moderator density.
- Boron concentration.
- Structure temperature, structural motion (rarely modeled).

Regardless of the model of feedback, reactivity is given by:

$$\rho(t) = \rho_{ex}(t) - \rho_b + \rho_{fb}(t), \quad (6)$$

in which $\rho_{ex}(t)$ is the external reactivity (usually due to control rod/blade movement), $\rho_b$ is the bias reactivity (used to enforce a desired initial reactivity $\rho^0 = \rho(t^0)$, which is zero for most transients), and $\rho_{fb}(t)$ is the reactivity feedback term. From here forward superscript 0 denotes initial/nominal quantities.

Point kinetics represent the neutron population in the reactor using a single scalar (a “lumped parameter” approach), but the associated thermal hydraulic model may be either lumped or spatially distributed. A lumped model does not divide the thermohydraulic compositions (fuel, moderator, etc.) into individual regions, as done in a spatially distributed model. Additionally, feedback models can be either separable, or coupled (“tabular,” or “multidimensional”). In a separable model, the change in feedback reactivity due to a change in a region’s temperature/density/boron concentration is independent of all other regions’ thermohydraulic states. One can postulate a coupled model in which the entire thermohydraulic state vector of the system affects the feedback reactivity, with each region’s state’s individual contributions being dependent on the thermohydraulic state of the system. Such model would be prohibitively expensive to implement, and so instead, coupled models are implemented as weighted averages of each composition’s thermohydraulic state variables affecting the feedback reactivity together, and the averages’ individual contributions being dependent on the thermohydraulic state of the system. This is the model presented below.

The spatially distributed separable reactivity feedback model is given by Eq. (7). The spatially distributed coupled feedback model, as described above, is given by Eq. (8). With $N_k$ being the number of regions in a composition $k$, the lumped models can be written similarly, with $N_k = 1$ for all $k$. Equations (7) and
represent only the thermal feedback, but additional terms can be naturally added to account for other feedback thermohydraulic variables, as listed above.

\[
\rho_b(t) = \sum_{k=1}^{K_s} \sum_{r=1}^{N_k} \alpha_{T,r,k} T_{r,k}(t) + \sum_{k=1}^{K_s} \sum_{r=1}^{N_k} W_{T,r,k} R_{T,r,k}(T_{r,k}(t)), \tag{7}
\]

\[
\rho_b(t) = R \left( \sum_{r=1}^{N_r} W_{T,r,1} T_{r,1}(t), \ldots, \sum_{r=1}^{N_r} W_{T,r,K_r} T_{r,K_r}(t) \right). \tag{8}
\]

Here, for region \( r \) of composition \( k \), \( T_{r,k}(t) \) is the temperature (if \( \rho_b \) is present) or its deviation from nominal, \( \alpha_{T,r,k} \) is the thermal feedback coefficient, and \( R_{T,r,k}(T_{r,k}(t)) \) is the nonlinear thermal feedback function. In Eq. (7), \( W_{T,r,k} \) is the nonlinear thermal feedback weighting factor, and in Eq. (8) it is the thermal feedback weighting factor to be used for the weighted average calculation for coupled feedback. \( R_{T,r,k}(T_{r,k}(t)) \) is the nonlinear thermal feedback function, and \( R(\ldots) \) is the \( K_r \)-dimensional coupled feedback function; both of these are normally nonlinear.

The first and second terms in Eq. (7) model the linear and nonlinear feedbacks, respectively. Generally, the separable contribution of a state variable is modeled either through a linear or a nonlinear term, but not both. For separable feedback, RELAP5-3D utilizes a form of Eq. (7) that treats moderator temperature linearly, moderator density nonlinearly (with the same nonlinear feedback shape function \( R_T(T_{r,k}) \) used for all thermohydraulic volumes), fuel temperature either linearly or nonlinearly, and boron concentration through control variables (which may be linear or nonlinear). Both the linear and nonlinear feedbacks will be represented in section 3, but only the linear feedback is used in the benchmark problem in section 5.

The original bond graph representation of point kinetics was based on a complete lumped parameter representation, similar to Eq. (7) with \( N_k = 1 \) \cite{Tyblec1981}. Because only fully linear models could be processed at the time, a significant simplification was made to the point kinetics model: in Eq. (3a), the \( \rho(t) P(t) \) term was replaced with \( \rho(t) P^0 \) to yield:

\[
\frac{d}{dt} P(t) = \frac{\rho(t)}{\Lambda} P^0 - \frac{\beta}{\Lambda} P(t) + \sum_{m=1}^{M} \lambda_m \tilde{C}_m(t). \tag{9}
\]

Equation (9) linearizes the problem, but introduces a significant inaccuracy into it: the prompt neutron generation term is now only accurate if the neutron population does not vary appreciably from nominal. Such approximation is sufficient for many sensitivity and stability studies, but cannot be used for super prompt-critical transients, transients that start from low power (like rod ejection from a hot zero power core, a standard design basis accident), and multiple others. The bond graph representation in this paper instead represents Eqs. (3), and so this limitation is removed.

2.2. Simplified thermal hydraulic equations

The benchmark problem discussed in section 5 is a pressurized water reactor (PWR) rod ejection transient. Typically, when modeling a PWR for an accident scenario, it is necessary to account for fluid compressibility and potential two-phase flow that may occur in the core. This is the primary task of thermal hydraulic systems codes. Such physical effects have previously been represented using bond graphs, using “convection bonds” \cite{Brown1991}. However, the transient addressed here is very short and fast; there typically isn’t the time for the core to depressurize or the coolant to flash under such conditions. For these reasons, and for clarity, we will omit these effects here. The axial mass flow rate will be assumed constant, and the coolant density will be assumed to only be the function of temperature. Additionally, we will ignore axial diffusion, which is consistent with RELAP5-3D assumptions. The 1D thermal energy advection equation then becomes:

\[
\frac{\partial}{\partial t} h_v(t, z) = -\frac{\partial}{\partial z} \left[ V(T) h_v(t, z) \right] + \dot{u}''(t, z), \tag{10}
\]
in which \( h_w(t, z) \), \( V(T) \) and \( u^m(t, z) \) are the fluid volumetric enthalpy density, axial velocity and heat rate density, respectively. With a known constant mass flow rate \( \dot{m} \), coolant density \( \rho_w(T) \), and coolant flow area \( A_w \), \( V(T) \) becomes:

\[
V(T) = \frac{\dot{m}}{\rho_w(T) A_w}.
\]

The heat rates are due to \( \gamma \)-heating of the fluid and its convective cooling of the fuel rods. To discretize the flow into \( N_i \) axial regions, we integrate Eq. (10) over axial region \( i \) of height \( \Delta z_i \) (\( i \) increases upward) to obtain:

\[
\frac{d}{dt} H_{w,i}(t) = \dot{H}_{w,i}(t) - \dot{H}_{w,i-1}(t) + \dot{Q}_w(t) + \dot{Q}_k(t),
\]

in which \( H_{w,i}(t) \) is the enthalpy in the coolant, \( \dot{H}_{w,i}(t) \) is the rate of upward enthalpy advection, \( \dot{Q}_w(t) \) and \( \dot{Q}_k(t) \) are the clad cooling and coolant \( \gamma \)-heating rates, respectively. All of these quantities are for coolant axial region \( i \).

Using the upwind differencing scheme, with \( T_{w,i} \) as the coolant axial region’s average temperature, and \( h_w(T_{w,i}) \) as the region’s specific enthalpy as a function of temperature (at operating pressure; a tabulated quantity), \( \dot{H}_{w,i}(t) \) can be written as:

\[
\dot{H}_{w,i}(t) = \dot{m} h_w(T_{w,i}) = \rho_w(T_{w,i}) V(T_{w,i}) h_w(T_{w,i}) A_w.
\]

The two formulations in Eq. (13) are equivalent. For enthalpy inflow rate into the core \( \dot{H}_{w,0}(t) \), a known inlet temperature \( T_{w,0}(t) \) can be used with Eq. (13).

In general, even for a single-phase flow, it is not possible to express the temperature in terms of volumetric enthalpy density, because the \( h_w(T) \) dependence is not necessarily strictly monotonic. However, for the nearly-incompressible assumptions made above to hold, the fluid’s density must not vary rapidly with temperature. This is the case when the fluid is sufficiently far from saturation temperature. Under these conditions, \( h_w(T) \) becomes strictly monotonic, and therefore invertible, and \( T(h_w) \) can be expressed; let us denote this function \( T_{h_w}(h_w) \). The average axial region \( i \) coolant temperature \( T_{w,i} \) is therefore related to the region’s enthalpy through:

\[
T_{w,i}(H_{w,i}) = T_{h_w}(H_{w,i}),
\]

with \( V_{w,i} \) as the coolant volume at axial region \( i \).

For solid materials, such as the fuel, specific thermal energy is typically not tabulated; instead, temperature and internal energy are related through volumetric heat capacity \( c_v(T) \), which is defined as:

\[
\frac{d}{dT} u_{v,k}(T) = c_{v,k}(T) = \rho_k(T) c_{p,k}(T),
\]

in which, for a material \( k \) at temperature \( T \), \( u_{v,k}(T) \) is the volumetric thermal energy density, and \( \rho_k(T) \) and \( c_{p,k}(T) \) are the density and specific heat capacity, respectively. \( \rho_k(T) \) and \( c_{p,k}(T) \) are the normally tabulated properties. Assuming \( u_{v,k}^0 = u_{v,k}(T^0) \), we can use separation of variables to solve Eq. (15):

\[
u_{v,k}(T) = \int_{T^0}^T dT' c_{v,k}(T') + u_{v,k}^0.
\]

Because internal energy is not a directly measurable property, and temperature is, \( u_{v,k}^0 \) may be set to zero by convention. \( c_{v,k}(T) \) is always positive, therefore the integral in Eq. (16) is strictly monotonically increasing. This makes \( u_{v,k}(T) \) strictly monotonic, and therefore invertible; its inverse is the temperature as a function of thermal energy \( T_{u_{v,k}}(u_{v,k}) \). For a material \( k \) region \( r \), let \( V_{k,r} \) be its volume, \( T_{k,r} \) its average temperature, and \( U_{k,r} \) the thermal energy in it. Note, that because more than one fuel pin is being modeled, when calculating solid regions’ volumes, a single region’s volume \( V_{k,r}^1 \) must be multiplied by a scaling factor
\( N_{\text{pi}} \) to yield \( V_{k,r} \). \( N_{\text{pi}} \) is normally the number of fuel pins, the same in every region, but may vary to account for partially-occupied regions. \( T_{k,r}(U_{k,r}) \) is then given by:

\[
T_{k,r}(U_{k,r}) = T_{u,k} \left( \frac{U_{k,r}}{V_{k,r}} \right).
\] (17)

For simplicity, in many codes, including RELAP5-3D, the gap conductance model is sometimes substituted with a thin, poorly-conducting material. The gap and clad are sufficiently thin to not require radial discretization for a simplified analysis. However, because of the wide temperature range across a ceramic fuel pellet’s radius, and the strong temperature dependence of its thermal properties, the pellet must be discretized radially.

Let the subscripts \( f \), \( g \) and \( c \) refer to fuel, gap and clad quantities, respectively, and the indices \( i \) and \( j \) index the regions axially and (for fuel only) radially, respectively. Discretizing the fuel pellet radius \( r_f \) into \( N_{fj} \) concentric circular cells, \( j \) increases outward from 1 to \( N_{fj} \). This way, \( N_f = N_{fj} N_i \) distinct fuel regions exist in the system. The index \( r \) above is then replaced by the pair \( (j,i) \) in fuel. Again neglecting axial heat diffusion and integrating over the corresponding axial regions yields the heat diffusion equations for the fuel, the gap and the clad:

\[
\frac{d}{dt} U_{fj,i}(t) = - \frac{T_{fj,i} - T_{fj+1,i}}{R_{fj,i}(T_{fj,i}, T_{fj+1,i})} + \dot{Q}_{fj,i}(t) \quad \text{if } j = 1,
\]

\[
\frac{d}{dt} U_{g,i}(t) = - \frac{R_{fg,i}(T_{g,i}, T_{c,i})}{R_{g,i}(T_{fN_{fj},i}, T_{g,i})} \left( T_{fN_{fj},i} - T_{g,i} \right) - \frac{T_{g,i} - T_{c,i}}{R_{gc,i}(T_{g,i}, T_{c,i})} + \dot{Q}_{g,i}(t) \quad \text{if } 1 < j < N_{fj},
\] (18a)

\[
\frac{d}{dt} U_{c,i}(t) = - \frac{R_{cw,i}(T_{c,i}, T_{w,i})}{R_{gc,i}(T_{g,i}, T_{c,i})} \left( T_{g,i} - T_{c,i} \right) - \frac{T_{c,i} - T_{w,i}}{R_{cw,i}(T_{c,i}, T_{w,i})} + \dot{Q}_{c,i}(t)
\] (18c)

in which \( \dot{Q}_{fj,i}(t) \) is the fission heat rate. The \( R_{fj,i}, R_{fg,i}, R_{gc,i} \) and \( R_{cw,i} \) terms are conductive and conductive+convective cylindrical thermal resistances between the corresponding regions. Thermal resistances are used here to quantify the heat transfer rates between the regions, by assuming that they are proportional to the differences between the corresponding regions’ average temperatures. To calculate the conductive resistances, within a radial cell, the properties are evaluated at the cell’s average temperature, and the temperature spatial derivative at the cell boundary is approximated by a finite difference between the cell’s midpoint and boundary. This is similar to the RELAP5-3D spatial discretization, which uses pure finite difference instead, and evaluates properties at mesh points, instead of at cell averages. Such approximation is also equivalent to the exact thermal resistance for a thin cylindrical shell without internal heat generation; heat generation is present, therefore sufficiently thin cell have to be used. These approximations yield the
following resistances:

\[
R_{fj,i} (\bar{T}_{fj,i}, T_{fj+1,i}) = \frac{1}{2\pi r_{fj} N_{pi} \Delta z_i} \left[ \frac{\Delta r_{fj}}{k_{fj}(\bar{T}_{fj,i})} + \frac{\Delta r_{fj+1}}{k_{fj}(\bar{T}_{fj+1,i})} \right], \tag{19a}
\]

\[
R_{fg,i} (\bar{T}_{fN_{fj,i}}, T_{g,i}) = \frac{1}{2\pi r_{f} N_{pi} \Delta z_i} \left[ \frac{\Delta r_{g}}{k_{g}(\bar{T}_{g,i})} + \frac{\Delta r_{g+1}}{k_{g}(\bar{T}_{g+1,i})} \right], \tag{19b}
\]

\[
R_{gc,i} (\bar{T}_{g,i}, T_{c,i}) = \frac{1}{2\pi r_{g} N_{pi} \Delta z_i} \left[ \frac{\Delta r_{c}}{k_{c}(\bar{T}_{c,i})} + \frac{1}{h(T_{w,i})} \right], \tag{19c}
\]

\[
R_{cw,i} (\bar{T}_{c,i}, T_{w,i}) = \frac{1}{2\pi r_{c} N_{pi} \Delta z_i} \left[ \frac{\Delta r_{c}}{k_{c}(\bar{T}_{c,i})} \right], \tag{19d}
\]

in which \(r_{fj/g/c}\) are the corresponding radial cells’ outer radii, and the \(\Delta r\) terms are their thicknesses. \(k(T)\) are the thermal conductivities, and \(h(T_{w,i})\) is the convective heat transfer coefficient (HTC). The second term in Eq. (18c) is the same as \(\dot{Q}_c^c(t)\) in Eq. (12).

The feedback modeled by Eqs. (7) and (8) normally uses radially averaged fuel temperatures, and not individual cells’ temperatures. To calculate the axial region \(i\) radially averaged fuel temperature \(\bar{T}_{fi}\), we make use of volume fractions \(f_{vj}\):

\[
f_{vj} = \frac{r_{fj}^2 - r_{fj-1}^2}{r_{f}^2}, \tag{20}
\]

in which \(r_{f0} = 0\) is used. \(\bar{T}_{fi}\) is a volume fraction-weighted sum of individual cells’ temperatures:

\[
\bar{T}_{fi} = \sum_{j=1}^{N_{fj}} f_{vj} T_{fj,i}. \tag{21}
\]

Lastly, the coolant gamma and fuel fission heat rates \(\dot{Q}^\gamma_i(t)\) and \(\dot{Q}^f_{j,i}(t)\) are simply fractions of the core power \(P(t)\), given by:

\[
\dot{Q}^\gamma_i(t) = \gamma_i P(t), \tag{22a}
\]

\[
\dot{Q}^f_{j,i}(t) = S_{j,i} P(t), \tag{22b}
\]

in which in which \(\gamma_i\) and \(S_{j,i}\) are the fractions of total reactor power as coolant axial region \(i\) gamma heat rate and fuel subcell \((j,i)\) fission heat source, respectively. Assuming radially uniform in-fuel heat generation, \(S_{j,i}\) can be expressed in terms of axial fuel fission heat source fraction \(S_i\) and volume fractions:

\[
S_{j,i} = f_{vj} S_i. \tag{23}
\]

The source fractions must add up to one:

\[
\sum_{\text{all } j,i} S_{j,i} + \sum_{\text{all } i} \gamma_i = 1. \tag{24}
\]

Together, Eqs. (11)–(14) and (17)–(22) model the thermal hydraulics of a one-dimensional, nearly incompressible flow typical of a coarse systems-level representation of a PWR core. Their bond graph representations, together with the nonlinear PKEs which model the core neutronics, are given in section 3.

3. Bond graph representation of PKEs with spatially distributed feedback

This section expects a certain level of familiarity with the bond graph formalism; Sosnovsky and Forget (2013) and Brown (2007) are the recommended references. The section summarizes the bond graph representation for the physics outlined in section 2 and may also be skipped without loss of continuity.
The bond graph representation of a system of ODEs, like the ones derived in section 2, starts by matching the ODEs with appropriate bond graph element equations (Sosnovsky and Forget, 2013). As specified in Tylee (1981), the PKEs are represented as follows: total reactor power is both a momentum and a flow, and the adjusted precursor concentrations \( \tilde{C}_m^*(t) \) are displacements. The rates of change of power and adjusted precursor concentrations, as well as individual contributions to them, are therefore efforts and flows, respectively. Reactivity is an effort which is used to modulate a resistor; bias reactivity, which is a constant but unknown quantity (solved for during the steady state search, and kept constant throughout the transient), is a displacement and an effort (Tylee (1981) does not use a bias reactivity).

The bond graph representations of incompressible thermal hydraulic models in literature normally use internal energy, and not enthalpy, as the displacement on thermal capacitors (Tylee, 1983; Karnopp, 1978); because the two quantities are similar, we will use enthalpy as the displacement instead. The various rates of change of enthalpy, as well as contributions to it, are therefore flows, and the temperatures are efforts. Conduction has been modeled with bond graphs in some detail (Sosnovsky et al., 2010; Sosnovsky, 2010); in it, internal energy is again a displacement, heat rates are flows, and temperature, again, an effort.

The feedback equations derived in section 2 supported an arbitrary number of compositions \( K_c \); below, a thermal feedback only from the coolant \( w \) and the fuel \( f \) is represented. The resulting representations can be naturally extended to support more compositions.

Using the above conventions, we can proceed to construct the bond graph representations of the equations in section 2.

Equation (5a) is a rate balance equation, which in the bond graph context means a balance of efforts. This is enforced by a 1-junction. The rate of change of power is a rate of change of a momentum, which is the effort supplied to an inertial element. Reactor power is also an effort, so the first term in Eq. (5a) can be viewed as a contribution to a rate of change of a momentum proportional to an effort; the proportionality coefficient is a combination of a function \( \rho(t)/\Lambda \) and a constant \( \beta/\Lambda \). These two components may be represented as an effort-modulated (reactivity is an effort) resistor and a constant coefficient resistor, respectively. Lastly, the second term in Eq. (5a) is a sum of individual contributions to rate of change of a momentum (power), which can be enforced by another 1-junction.

Combined, these equivalences result in the bond graph representation in Figure 1. The transformers and 0-junctions present in it are discussed below.

---

**Figure 1: Power Equation Bond Graph Representation**

The constituent expressions of the elements in Figure 1 are given by:

\[
I_P = 1, \quad (25a)
\]

\[
R_P = \frac{\beta}{\Lambda}, \quad (25b)
\]

\[
R_{\rho P}(\varepsilon_{\rho}) = \varepsilon_{\rho}, \quad (25c)
\]

in which \( I_P \) is the parameter of the power inertial element \( I_P \), \( R_P \) is the resistance of the constant power resistor \( R_P \), \( R_{\rho P}(\varepsilon_{\rho}) \) is the resistance of the functional (reactivity) modulated power resistor \( MR_{\rho P} \), and \( \varepsilon_{\rho} \) is the reactor reactivity, the modulating effort for \( MR_{\rho P} \). Figure 1 is very similar to the corresponding representation in Tylee (1981), with the exception of \( MR_{\rho P} \), which is a new (modulated, and therefore...
nonlinear) addition, replacing a linear transformer. Tylee could not use modulated (nonlinear) resistors, because they prevented the model from being fully linear, and therefore able to be processed by ENPORT.

The adjusted delayed neutron precursor equations (5b) are represented the same way as in Tylee (1981); this representation is given in Figure 2. The $1_\beta$-junction in it is the same element as in Figure 1, through which the two physics are connected.

The capacitance and resistance here are chosen in such a way as to enforce the appropriate decay rate-based contributions into the $1_\beta$-junction in Figure 1 while also representing Eq. (5b). They are given by:

$$C_m = \frac{\lambda_m \beta_m}{\Lambda},$$  \hspace{1cm} (26a)  

$$R_m = \frac{\beta_m}{\Lambda},$$  \hspace{1cm} (26b)

in which $C_m$ and $R_m$ are the parameters of the precursor family $m$’s capacitor $C_m$ and resistor $R_m$, respectively.

Reactivity, which modulates $MR_{P_P}$ in Figure 1, is given by Eq. (6); as stated above, it is an effort, algebraically computed by a 1-junction. The external reactivity contribution $\rho_{ex}(t)$ is only time-dependent, and therefore supplied by a time-modulated source of effort element. Bias reactivity is an effort supplied by a capacitor (equal to the capacitor’s displacement). The feedback reactivity term is either a sum of efforts from individual contributions (separable feedback, Eq. (7)), or a single effort from a multidimensional function evaluation (coupled feedback, Eq. (8)). A fuel region’s average fuel temperature (Eq. (20)) is a weighted sum of efforts, which can be accomplished by a single 1-junction and one linear transformer element per radial cell.

In a linear separable feedback model, all contributing efforts are multiplied by constant feedback coefficients. Because both the weighting volume fractions for the individual fuel temperatures and the feedback coefficients are constant multipliers, a single linear transformer element (one per contributing region) can represent both these coefficients. Signal bonds must be used instead of full bonds, because reactivity does not directly contribute energy to the system: it only affects the rate of change of power through the $MR_{P_P}$ resistor in Figure 1. The reactivity with linear separable feedback representation is given in Figure 3. With the exception of the nonlinear resistor $MR_{P_P}$ and the bias reactivity’s capacitor $C_{\rho, b}$, this representation is identical to the one in Tylee (1981).

In a nonlinear separable feedback model, nonlinear functions of the contributing efforts are evaluated.
Nonlinear 2-port resistive elements in can be used to enforce these functions. Such model is represented in Figure 4

![Figure 4: Reactivity with Nonlinear Separable Feedback Bond Graph Representation](image)

In a coupled feedback model, weighted averages (with constant weighting coefficients) for each composition (here, for coolant and fuel) are computed prior to evaluating the overall reactivity feedback. Such averages, as well as individual contributions to them, are efforts, summed by 1-junction. As in the linear separable feedback model, the weighting is implemented through linear transformer elements, one per contributing region. These efforts are then supplied to a multiport nonlinear resistor, which outputs the coupled reactivity feedback term — also an effort. Because in this example there are two inputs and one output, this multiport resistor is an R3 element. The reactivity with coupled feedback representation is given in Figure 5

![Figure 5: Reactivity with Coupled Feedback Bond Graph Representation](image)

In all three of the above reactivity representations, the following constituent expressions are used:

\[
e_{TSE_{\rho,ex}}(t) = \rho_{ex}(t),
\]

\[
C_{\rho,b} = 1,
\]

in which \(e_{TSE_{\rho,ex}}(t)\) is the effort output by the external reactivity time-modulated source of effort \(TSE_{\rho,ex}\) and \(C_{\rho,b}\) is the parameter of the bias reactivity element \(C_{\rho,b}\).

In the linear separable feedback representation (Figure 3), the following constituent expressions are used:

\[
\mu_{\rho w,i} = \frac{1}{\alpha_{w,i}},
\]

\[
\mu_{\rho f,j,i} = \frac{1}{\rho_{f,j,i}},
\]

in which \(\mu_{\rho w,i}\) and \(\mu_{\rho f,j,i}\) are the parameters of the coolant and fuel thermal feedback transformers \(TF_{\rho w,i}\) and \(TF_{\rho f,j,i}\), respectively. There is one coolant and \(N_{f,j}\) fuel thermal feedback transformers per region \(i\).

In the nonlinear separable feedback representation (Figure 4), the R2 elements have only one constituent expression each — because of the causality and the active bonds used, the other output is zero. The fuel thermal feedback transformers now only represent the radial averaging coefficients. The constituent
expressions are:

$$\varepsilon_{R2w,i}(e_{T,i,w}) = W_{T,i,w} R_{T,i,w} (e_{T,i,w}),$$ (29a)
$$\varepsilon_{R2f,i}(e_{T,i,f}) = W_{T,i,f} R_{T,i,f} (e_{T,i,f}),$$ (29b)
$$\mu_{fj,i} = \frac{1}{f_{ij}},$$ (29c)

in which \(\varepsilon_{R2w,i}(e_{T,i,w})\) and \(\varepsilon_{R2f,i}(e_{T,i,f})\) are the efforts output by the coolant and fuel thermal feedback resistors \(R_{2w,i}\) and \(R_{2f,i}\), respectively. \(e_{T,i,w}\) and \(e_{T,i,f}\) are the efforts (temperatures) supplied to these resistors from the corresponding thermohydraulic 1-junctions. \(\mu_{fj,i}\) are the parameters of the radial averaging transformers \(TF_{fj,i}\).

In the coupled feedback representation (Figure 5), the R3 element has only one constituent expression — again, because of the causality and the active bonds used. The other outputs are zero. The elements' constituent expressions are:

$$\mu_{pw,i} = \frac{1}{W_{T,i,w}},$$ (30a)
$$\mu_{fj,i} = \frac{1}{f_{ij} W_{T,i,f}},$$ (30b)
$$\varepsilon_{R3w}(e_{T,w}, e_{T,f}) = R(e_{T,w}, e_{T,f}),$$ (30c)

in which \(\mu_{pw,i}\) and \(\mu_{fj,i}\) are the parameters of the coolant and fuel thermal feedback weighting transformers \(TF_{pw,i}\) and \(TF_{fj,i}\), respectively. There is one coolant and \(N_{fj}\) fuel thermal feedback transformers per region \(i\). The transformers' outputs are summed at the 1\(_w\) and 1\(_f\) junctions; these sums are then supplied as efforts \(e_{T,w}\) and \(e_{T,f}\) to the R3\(_w\) feedback coupling element. This element outputs the effort \(\varepsilon_{R3w}(e_{T,w}, e_{T,f})\).

This completes the bond graph representation of the nonlinear PKEs from section 2.1. The nearly incompressible thermal hydraulic equations from section 2.2 are represented below.

Equation (12) is a rate balance equation; because enthalpy is a displacement, and its rate of change a flow, the equation is a balance of flows, which is enforced by a 0-junction. Coolant temperature (an effort) is a function of enthalpy (a displacement) through Eq. (14), which is enforced by a capacitor. The upwind enthalpy flow rate specified in Eq. (13) is a flow which is a function of temperature (an effort), and this flow is simultaneously removed from region \(i\) and added to region \(i + 1\). This makes an R2 resistive element with an active bond on the outlet port a good candidate for setting the flow. For core inlet enthalpy flow rate, a time-modulated source of flow can be used instead; similarly, a flow sink can be used at the outlet. Lastly, the gamma heat rate in Eq. (22a) is simply proportional to power (an effort in Figure 1), which, like the thermal feedback, can be easily enforced through a linear transformer with two signal bonds.

The last term in Eq. (12), \(\dot{Q}_c(t)\), is represented below, together with other radial heat transfer terms in Eqs. (18). Leaving it out for now, the advection physics get represented through Figure 6.

In Figure 6 the following constituent expressions are used:

$$\varepsilon_{C_{w,i}}(q_{w,i}) = T_{h,c}(\frac{q_{w,i}}{V_{w,i}}),$$ (31a)
$$f_{R2_{w,i}}^{in/out}(e_{T,w,i}) = \dot{m} h_w(e_{T,w,i}),$$ (31b)

in which \(\varepsilon_{C_{w,i}}(q_{w,i})\) is the effort (temperature) output by the coolant capacitor \(C_{w,i}\) as a function of displacement (enthalpy) \(q_{w,i}\) and \(f_{R2_{w,i}}^{in/out}(e_{T,w,i})\) is the inlet and outlet flow output by the advective two-port resistor \(R2_{w,i}\). Because of the signal bond, it is only dependent on the effort delivered to the first port.

The gamma heat rate is provided to 0\(_w,i\) from 1\(_p\) through a linear transformer \(TF_{pw,i}\), shown on Figure 1. This transformer’s parameter is given by:

$$\mu_{pw,i} = \gamma_i.$$ (32)
Lastly, we can represent the fuel, gap and clad radial heat transfer equations (18). Diffusive heat transfer has been represented many times, including in a slab reactor in Sosnovsky et al. (2010); the heat transfer rates (individual terms of Eqs. (18)) are modulated single-port resistors which are supplied a difference in neighboring efforts (imposed through 1-junctions) and the temperatures are imposed onto 0-junctions through nonlinear capacitors (same as for coolant, but using Eq. (17) instead). The representation of the in-fuel radial heat transfer (Eq. (18a)) is shown in Figure 7, the fuel, gap and clad radial heat transfer (Eqs. (18b) and (18c)) representation is shown in Figure 8.

Using $k$ to denote fuel ($fj$), gap ($g$) and clad ($c$) subscripts, the capacitive elements’ constituent equations for Figures 7 and 8 are:

$$e_{C_{k,i}}(q_{k,i}) = T_{uv} \left( \frac{q_{k,i}}{V_{k,i}} \right),$$  \hspace{1cm} (33)

in which $e_{C_{k}}(q_{k,i})$ is the effort (temperature) output by the capacitor $C_{k,i}$ as a function of displacement (thermal energy) $q_{k,i}$. The resistive parameters of the modulated resistors $MR_{fj,i}$, $MR_{fg,i}$, $MR_{gc,i}$ and $MR_{cw,i}$ are the resistances from Eqs. (19), which, in terms of modulating efforts (supplied from the corresponding
0-junctions), become:

\[
R_{fj,i}(e_{fj,i}, e_{fj+1,i}) = \frac{1}{2\pi r_f N_{pi} \Delta z_i} \left[ \frac{\Delta r_{fj}}{k_f(e_{fj,i})} + \frac{\Delta r_{fj+1}}{k_f(e_{fj+1,i})} \right], \tag{34a}
\]

\[
R_{fg,i}(e_{fN_{fj,i}}, e_{g,i}) = \frac{1}{2\pi r_f N_{pi} \Delta z_i} \left[ \frac{\Delta r_{fg}}{k_g(e_{g,i})} + \frac{\Delta r_{g}}{k_g(e_{g,i})} \right], \tag{34b}
\]

\[
R_{ge,i}(e_{g,i}, e_{c,i}) = \frac{1}{2\pi r_g N_{pi} \Delta z_i} \left[ \frac{\Delta r_{ge}}{k_c(e_{c,i})} + \frac{\Delta r_{g}}{k_c(e_{c,i})} \right], \tag{34c}
\]

\[
R_{cw,i}(e_{c,i}, e_{w,i}) = \frac{1}{2\pi r_c N_{pi} \Delta z_i} \left[ \frac{\Delta r_{cw}}{k_c(e_{c,i})} + \frac{1}{h(e_{w,i})} \right], \tag{34d}
\]

The fission heat rate is provided to 0-junctions, \( R_{fj,i} \) from 1\( p \) through a linear transformer \( T_F p_{fj,i} \), shown on Figure 1. This transformer’s parameter is given by:

\[
\mu p_{fj,i} = S_{j,i}. \tag{35}
\]

This completes the representation of the simplified thermal hydraulic equations in section 2.2, and of the physics we were trying to represent. Note, that Figures 1–8 are all connected through appropriately indexed junctions. Section 3 illustrates a sample PWR rod ejection problem using this model. We may now proceed to discussing the code developed to process these (potentially highly nonlinear) bond graph representations, \textit{BGSolver}.

4. BGSolver: a nonlinear bond graph processing code

The advantage of using the bond graph process is the ability to use a bond graph processing code to construct a state derivative vector, and, optionally, its Jacobian (or Jacobian sparsity pattern), and integrate the resulting system or search for its steady state. \cite{Sosnovsky and Forget 2013} provides a review of existing bond graph processing codes. Because they are all, for various reasons, unfit for processing discretized nonlinear bond graph systems such as the one in section 3, a new bond graph processing code, \textit{BGSolver}, was developed. The code’s source and compiled MATLAB executables are now available online as open source software \cite{BGSolver}. \textit{BGSolver’s} algorithms were detailed in \cite{Sosnovsky and Forget 2013}; because the code is now released as open source software, this section discusses its capabilities and provides instructions for its use.

4.1. Supported bond graph systems

\textit{BGSolver} is a MATLAB-based bond graph processing code designed to process (sort, and, optionally, integrate) fully causal bond graph systems. The bonds’ causalities, with the exception of resistive elements with ambiguous causality, do not need to be specified explicitly. \textit{BGSolver} is able to process acausal bond graph systems composed of the bond graph elements listed in Table 1. In it: \( e \) and \( f \) are the effort and flow on the bonds connected to single-port elements; \( F(\ldots) \) and \( \mathbf{F}(\ldots) \) are scalar and vector functions of one or more variables, respectively; \( \mathbf{m} \) is a vector of modulating variables (may include time, state and bond variables); \( q \) and \( p \) are the displacement and momentum on corresponding storage elements, respectively; \( C, I, \mu, \) and \( R \) are the corresponding elements’ parameters; \( e_k \) and \( f_k \) are the effort and flow on port \( k \) of a multiport element, respectively; \( N \) is the number of ports on a multiport element; \( e_{to/from} \) and \( f_{to/from} \) are the effort and flow variables on the bonds pointing \( to/from \) 2-port elements; \( \mathbf{Y} \) and \( \mathbf{Y}(\ldots) \) are a constant matrix and a matrix function, respectively; \( \mathbf{B}_{in/out} \) are the input/output bond variable vectors for multiport resistors. The bond variable vectors are configured as follows: on a multiport resistor, each individual port has an input and an output bond variable, therefore, if \( e_k \) is an output, \( f_k \) must be an input, and vice versa.

Time-dependent sources are implemented as regular modulated source elements, with time as the only specified modulating variable. Signal bonds are not implemented directly in \textit{BGSolver}. Instead, modulating
Table 1: Bond graph elements supported by B2Solver

<table>
<thead>
<tr>
<th>Element type</th>
<th>Name</th>
<th>Constituent algebraic equation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE</td>
<td>Source of Effort</td>
<td>$e = \text{constant}$</td>
</tr>
<tr>
<td>SF</td>
<td>Source of Flow</td>
<td>$f = \text{constant}$</td>
</tr>
<tr>
<td>MSE</td>
<td>Modulated Source of Effort</td>
<td>$e = F(\vec{m})$</td>
</tr>
<tr>
<td>MSF</td>
<td>Modulated Source of Flow</td>
<td>$f = F(\vec{m})$</td>
</tr>
</tbody>
</table>
| C            | Capacitive element | $e = Cq$ or $e = F(q)$
| I            | Inertial element  | $f = Ip$ or $f = F(p)$
| MC           | Modulated Capacitive element | $e = F(\vec{m})q$ or $e = F(q, \vec{m})$ |
| MI           | Modulated Inertial element | $f = F(\vec{m})p$ or $f = F(p, \vec{m})$ |
| 1            | 1-junction        | $f_1 = \ldots = f_N$ and $\sum_{k=1}^{N} e_k = 0$ |
| 0            | 0-junction        | $e_1 = \ldots = e_N$ and $\sum_{k=1}^{N} f_k = 0$ |
| TF           | Transformer       | $e_{to} = \mu e_{from}$ and $f_{from} = \mu f_{from}$ |
| GY           | Gyrator           | $e_{to} = \mu f_{from}$ and $f_{from} = \mu e_{from}$ |
| MTF          | Modulated Transformer | $e_{to} = F(\vec{m}) e_{from}$ and $f_{from} = F(\vec{m}) f_{out}$ |
| MGY          | Modulated Gyrator | $e_{to} = F(\vec{m}) f_{from}$ and $f_{from} = F(\vec{m}) e_{out}$ |
| R            | Resistive element | $e = Rf$ or $e = F(f)$ or $f = F(e)$
| MR           | Modulated Resistive element | $e = F(\vec{m}) f$ or $e = F(f, \vec{m})$ or $f = F(e, \vec{m})$
| R2           | 2-port Resistive element | $\vec{b}_{out} = \vec{Y} \vec{b}_{in}$ or $\vec{b}_{out} = \vec{F}(\vec{b}_{in})$ with $\vec{b}_{out}, \vec{b}_{in} \in \mathbb{R}^2$, $\vec{Y} \in \mathbb{R}^{2 \times 2}$ and $\vec{F}(\ldots) \in \mathbb{R}^{2 \times 2}$ |
| MR2          | Modulated 2-port Resistive element | $\vec{b}_{out} = \vec{Y}(\vec{m}) \vec{b}_{in}$ or $\vec{b}_{out} = \vec{F}(\vec{b}_{in}, \vec{m})$ with $\vec{b}_{out}, \vec{b}_{in} \in \mathbb{R}^2$, $\vec{Y}(\ldots) \in \mathbb{R}^{2 \times 2}$ and $\vec{F}(\ldots) \in \mathbb{R}^{2 \times 2}$ |
| RN           | N-port Resistive element | $\vec{b}_{out} = \vec{Y} \vec{b}_{in}$ or $\vec{b}_{out} = \vec{F}(\vec{b}_{in})$ with $\vec{b}_{out}, \vec{b}_{in} \in \mathbb{R}^N$, $\vec{Y} \in \mathbb{R}^{N \times N}$ and $\vec{F}(\ldots) \in \mathbb{R}^{N \times N}$ |
| MRN          | Modulated N-port Resistive element | $\vec{b}_{out} = \vec{Y}(\vec{m}) \vec{b}_{in}$ or $\vec{b}_{out} = \vec{F}(\vec{b}_{in}, \vec{m})$ with $\vec{b}_{out}, \vec{b}_{in} \in \mathbb{R}^N$, $\vec{Y}(\ldots) \in \mathbb{R}^{N \times N}$ and $\vec{F}(\ldots) \in \mathbb{R}^{N \times N}$ |

a Regular and modulated inertial and capacitive elements also have associated differential equations: $f = dq/dt$ and $e = dp/dt$.

b Regular and modulated transformers, gyrators and 2-port resistors are required to have exactly one bond pointing to port 1 and one bond pointing from port 2 of the element.

c If a nonlinear constituent expression is used for a single-port regular or modulated resistive element, its causality must be explicitly specified.

d Regular and modulated multiport resistors’ ports’ causalities must always be explicitly specified.

(junction-to-modulated element, like the bond from $1_\rho$ to $MR_{\rho \rho}$ in Figure 3) signal bonds are modeled by explicitly specifying the list of modulating variables for every modulating element. Active bonds connected to regular ports (like the bonds connected to transformers in Figure 3) are modeled by replacing the element with a multiport resistor, making the active bonds regular full bonds, and outputting zeros on the ports to which the active bonds are connected. An example of this substitution applied to a moderator thermal feedback transformer from Figure 3 is given in Figure 9.

The transformer’s parameter $\mu_{\rho \rho, i}$ is given in Eq. (28a). The equivalent 2-port resistive element’s coeffi-
The coefficient matrix is given by:

\[
Y = \begin{bmatrix}
0 & 0 \\
1 & 0 \\
\mu_{\rho w,i} & 0
\end{bmatrix}
\] (36)

\textbf{BGSolver} is capable of processing all bond graph systems described in this section. The algorithms it uses for doing so are briefly outlined in the following section.

4.2 Bond graph processing algorithms

The bond graph process consists of 7 steps: 1) partial differential (or integro-differential) equation discretization, 2) state vector indexing, 3) casting (“representing”) the problem as a bond graph system, 4) formulation of the algebraic equations, 5) sorting of the algebraic equation, 6) state derivative formulation, and 7) state derivative integration. A bond graph processing code like \textbf{BGSolver} handles the problem-independent steps 4–7.

Step 4 is implemented by building a MATLAB symbolic vector. During step 5, the Symbolic Math Toolbox (SMT) algebraic \texttt{solve} function is used to sort the algebraic equation vector, which yields all of the bond variables. In step 6, the flows on capacitors and efforts on inertial elements are extracted from the bond variable vector, thereby forming the state derivative vector. In step 7, the state derivative vector is integrated, using either a built-in MATLAB time integrator like \texttt{ode15s}, or a manually-written time integrator.

The symbolic sort in step 5 is the bottleneck section of the code. It is significantly more complicated than a simple SMT \texttt{solve} if some of the elements' constituent expressions are not constant coefficients; this and other algorithmic details are given in (Sosnovsky and Forget, 2013).

Lastly, after a state derivative vector is implemented, it can be integrated, or solved for steady state using a nonlinear solver, such as MATLAB Optimization Toolbox (OT) \texttt{fsolve} function. If the problem becomes an eigenvalue problem at steady state, such as a steady state neutronics problem without a fixed external source, it is often possible to write a custom nonlinear solver that would close the system (normally by specifying steady reactor power) and solve a nonlinear system for the steady state and the eigenvalue:

\[
\bar{F}(\bar{x}, k_{eff}) = \bar{0},
\] (37)

in which \(\bar{x}\) is the state vector, \(\bar{F}(\ldots)\) is the \(d\bar{x}/dt\) vector together with the system-closing equation’s residual, and \(k_{eff}\) is the (reactor) eigenvalue. Such nonlinear eigenvalue solvers are generally less efficient than true (power iteration-based) eigenvalue solvers, but are relatively simple to implement, and are widely used for steady state searches in fully coupled multiphysics codes for reactor analysis, such as PRONGHORN (Bingham et al., 2012). In the representation in section 3, the bias reactivity plays the role of the eigenvalue; because it is an invariant parameter, the element of \(\bar{F}(\ldots)\) that corresponds to \(d\rho_b/dt\) can simply be replaced with the closing equation’s residual (steady power specification) for the steady state search. This is the method used to find the steady state solution for the benchmark problem in section 5. It should be noted
that because such custom steady state solvers are generally problem- or at least physics-specific, they are not distributed with BGSolver.

The next section summarizes the input that BGSolver requires.

4.3. Input specifications

A problem input to BGSolver consists of two parts:

1. Run options. This part of the input specifies the options that do not change the bond graph system itself, but change how it is processed. Such options include verbosity, time integrator options, nonlinear solver and post-processing options. They do not have to be input, all run options have default values.

2. Bond graph system. This part of the input specifies the bond graph system to be processed according to the run options. It includes:

   (a) Element list, including element types, and, if applicable, constituent expressions, numbers of ports, causalities on ports, and initial conditions.

   (b) Bond connectivity map, which lists all full bonds in the system, and which elements they connect.

The two parts can be provided in two ways: either as a “.BGSD” text file (Bond Graph System Descriptor), or as two MATLAB structures. The Bond Graph System Descriptor format was developed specifically for BGSolver, and is documented in detail in Sosnovsky (2013). The specifications for MATLAB structures are available within the MATLAB code itself. When running BGSolver with a .BGSD input, it is possible to supply a new run options structure, which overwrites the run options in the file. This is helpful for running a problem in several different configurations, which is convenient for convergence studies like the one in section 5.

When non-constant coefficient constituent expressions are provided, there are two ways to implement them: either by entering a closed-form MATLAB symbolic expression, if possible, or by providing a MATLAB function handle. Function handles are frequently table interpolations, which rely on external data that can be loaded from .mat files listed in the run options. This allows the processing of the same problem with different data files, without having to create a separate .BGSD for each configuration; in particular, this is relevant to spatial and point kinetics analysis, in which the data is frequently uncertain.

Step 7 of the bond graph process — the time integration of the state derivative vector — completes the bond graph process itself. BGSolver provides one additional operation: the post-processing. The post-processing here consists of evaluating a specified list of bond variables based on the time vector and the state array (one vector per state variable) that the time integration produces. Such operation is particularly useful if only a small number of post-processed variables are actually of interest; these typically include temperatures, local heat rates, neutron leakage rates, and similar parameters. Such parameters can then be analyzed further, but this is outside of BGSolver’s scope.

The next section summarizes the output that BGSolver generates.

4.4. Output specifications

BGSolver outputs at most four arrays:

1. The vector of time points that the time integrator required. These may be uniformly (if fixed time stepping was used) or nonuniformly (if adaptive time stepping was used) distributed.

2. The set of vectors of state variables, evaluated at the time points in the time vector.

3. The set of efforts evaluated during post-processing at the time points in the time vector. If no efforts were specified during post-processing, this return is empty.

4. The set of flows evaluated during post-processing at the time points in the time vector. If no flows were specified during post-processing, this return is empty.
Subsequent analysis of the problem is based on these outputs, and is outside the scope of BGSolver.

This completes the description of BGSolver in its open source release form. This version was previously used for a spatial kinetics BWR blade drop problem (Sosnovsky and Forget [2013]; section 6 summarizes its results for a simpler PWR rod ejection problem.

5. Benchmark problem

The benchmark problem addressed here is a modified version of the typical PWR core from RELAP5-3D sample decks collection ([The RELAP5-3D Code Development Team] 2005, typpwr.i deck distributed with the code). The problem’s physics and spatial discretization are detailed in section 2. The geometry and other parameters are summarized in section 5.1 and listed in detail in Appendix A.

Both the steady state solution and the implicit time stepping were run using a modified Newton’s method-based nonlinear solver, described in section 5.2. The transients and the observed convergence in time, as well as the comparison of BGSolver’s results with RELAP5-3D are discussed in sections 5.3 and 5.4.

5.1. Problem statement

The core’s geometry, point kinetics parameters, feedback coefficients, gamma and fission heat rate fractions and fuel, gap and clad material properties are the same as specified in typpwr.i, with the following modifications:

1. $N_i = 36,552$ for all axial segments in the benchmark; in the deck, the scaling factors vary.
2. Clad thickness $\Delta r_c = 0.57$ mm in the benchmark; in the deck, the clad is 0.1524 mm instead.
3. $\beta = 0.0065$ (the delayed neutron fraction for Uranium-235) is used in the benchmark; in the deck, $\beta/\Lambda = 297$ s$^{-1}$ is used. Keeping this $\beta/\Lambda$ gives $\Lambda = 2.18855 \times 10^{-5}$ s.
4. $N_fj = 10$ is used in the benchmark solution, with one gap and one clad cell, as in section 2.2, in the deck, a finite difference discretization with 17 points in the radial mesh is used.
5. $\dot{m} = 18,395.15$ kg s$^{-1}$, $p_{\text{core}} = 156,5478$ bar and $T_{w,0} = 550$ K are used in the benchmark; here $p_{\text{core}}$ is the core average operating pressure at which the coolant properties are evaluated as functions of temperature. These parameters are not available in the deck explicitly, because the deck models the full primary system, and so these parameters are not generally kept constant. These specific parameters were chosen as the steady state conditions from the deck.
6. Weisman correlation ([Weisman] 1959) is used to evaluate the convective HTC in the benchmark; in the deck, the HTC is not specified explicitly, and is instead computed by RELAP5-3D’s heat transfer module. The steady state error between the two HTCs is under 2%.
7. IAPWS-IF97 ([Wagner et al.] 2000) water properties are used in the benchmark; RELAP5-3D, by default, uses the IFC-67 (International Formulation Committee of the 6th International Conference on the Properties of Steam 1967) water properties.

The complete problem statement is given in Appendix A.

Two sets of initial conditions (ICs) are used: 1) the “hot zero power condition (HZP),” and 2) the “very hot zero power condition (VHZP).” The HZP IC is obtained by finding the steady state with the steady reactor power of $P^0 = 1$ W. The VHZP IC is obtained by first finding the steady state at nominal reactor power of $P^0 = 3600$ MW, followed by setting the initial transient power to $P^0 = 1$ W and setting the initial delayed neutron powers $\tilde{C}_m^0$ in equilibrium with this $P^0$:

$$\tilde{C}_m^0 = \frac{\beta_m}{\lambda_m \Lambda} P^0,$$  

(38)
from which $\tilde{C}_{m,0}$ can be obtained through Eq. (5b). The HZP IC essentially represents a steady shutdown core with hot inlet water. The VHZP IC is not a steady state: because the 1W power is negligible for a full core, the fuel temperature will not stay steady, and will be dropping due to the coolant moving at full velocity through the core. The VHZP IC is studied here to stress the code: because it emphasizes the effect of cooling on the short transient, it is generally more difficult for a code to simulate, compared to the HZP IC. The VHZP IC is less physically realistic.

The transient studied consists of a ramp injection of $\Delta \rho_{CR} = 2$ of reactivity in $\Delta t_{CR} = 1$ s, which corresponds to the external reactivity curve of:

$\rho_{ex}(t) = \begin{cases} 
0 & \text{if } t < 0 \text{s}, \\
\left(\frac{t}{\Delta t_{CR}}\right) \Delta \rho_{CR} & \text{if } 0 \text{s} \leq t < \Delta t_{CR}, \\
\Delta \rho_{CR} & \text{if } \Delta t_{CR} \leq t .
\end{cases}$

(39)

The transient is integrated up to 3 seconds.

5.2. Nonlinear solver details

In a previous study, MATLAB OT’s \texttt{fsolve} was used as the nonlinear solver. This function is a form of the Newton-Raphson (NR) method with a Trust-Region Dogleg globalization algorithm, and is very efficient, but can have problems with robustness, particularly for multiphysics (poorly scaled and conditioned) problems. For this reason, a new, more robust nonlinear solver was developed for this problem (and also for future use).

A conventional Newton-Raphson method solves the problem $\bar{\mathbf{F}}(\bar{x}) = \mathbf{0}$ with an initial guess $\bar{x}^0$ by solving the following equation for $\bar{x}^k$ at each (“Newton-Raphson”) iteration $k$:

$\mathbf{J}^{k-1}(\bar{x}^k - \bar{x}^{k-1}) = -\bar{\mathbf{F}}^{k-1}$, 

(40)

in which the superscript $k$ denotes the iteration index (starting with the initial guess $k = 0$), $\bar{\mathbf{F}}^{k-1} = \bar{\mathbf{F}}(\bar{x}^{k-1})$, and $\mathbf{J}^{k-1} = \mathbf{J}(\bar{x}^{k-1})$ is the Jacobian of $\bar{\mathbf{F}}(\bar{x})$ at $\bar{x} = \bar{x}^{k-1}$.

As suggested in Press et al. (2007), there exist “no good, general methods for solving systems of more than one nonlinear equation.” However, several ways to make the Newton-Raphson method more robust are outlined in Strang (2007). The following techniques are used here to improve the conventional Newton-Raphson method:

1. A fixed-point Picard iteration is added to jumpstart the method: a specified number $N_P$ of Picard iterations (Newton-Raphson-like iterations without updating $\mathbf{J}^0$ between iterations) are run prior to running the Newton-Raphson iterations.

2. A Krasnosel’skii damping factor $\lambda_P$ and a Krasnosel’skii-like damping factor $\lambda_{NR}$ are added to the Picard and Newton-Raphson iterations, respectively.

Algorithm \ref{alg:nonlinear} provides the pseudocode for the resulting nonlinear solver. In it, $N_{NR}$ is the maximum number of NR iterations to run. $N_P = 0$ and $\lambda_{NR} = 1.0$ reduces the solver to the conventional Newton-Raphson method in Eq. (40).

The bulk of the cost of Algorithm \ref{alg:nonlinear} is in the evaluation of the Jacobian and the solutions of the linear system to update the solution vector. In this benchmark, MATLAB’s \texttt{numjac} function was used to numerically estimate the Jacobian (uses the algorithm from Salane (1986)), and direct Gaussian elimination was used to solve the linear systems. For a large problem, Algorithm \ref{alg:nonlinear} is still applicable, but a Jacobian-free method may be preferable (Knoll and Keyes, 2004).

The addition of the step-reducing $\lambda_P$ and $\lambda_{NR}$ factors helps convergence for two reasons: 1) they reduce the probability of overshooting the solution and converging onto a local minimum of the residual, and 2) by restricting the size of the step, the method may be prevented from stepping outside of the domain
For this benchmark problem: as discussed in section 2.2, the nearly incompressible model only works if appropriate extrapolation procedures are used. This is particularly relevant to the water property evaluation domain boundary, an overstep typically causes an immediate failure of the state derivative evaluation, unless

\[ \text{Require:} \ F(\bar{x}), \ J(\bar{x}), \ \lambda_P, \ \lambda_{NR}, \ N_P, \ N_{NR} \]

1: \( \mathbf{F}^0 \leftarrow \mathbf{F}(\mathbf{X}^0) \)
2: \( \mathbf{J}^0 \leftarrow J(\mathbf{X}^0) \)
3: for \( i_P = 1, N_P \) do
4: \( \mathbf{X}_{full}^{i_P} \leftarrow \text{SOLVE} \left( \mathbf{J}^0 \left( \mathbf{X}_{full}^{i_P} - \mathbf{X}_{NR}^{i_P-1} \right) = -\mathbf{F}^{i_P-1} \right) \)
5: \( \mathbf{X}^{i_P} \leftarrow (1 - \lambda_P) \mathbf{X}_{full}^{i_P} + \lambda_P \mathbf{X}_{full}^{i_P} \)
6: \( \mathbf{F}^{i_P} \leftarrow \mathbf{F}(\mathbf{X}^{i_P}) \)
7: end for
8: \( \mathbf{X}^0 \leftarrow \mathbf{X}^{N_P} \)
9: \( \mathbf{F}^0 \leftarrow \mathbf{F}(\mathbf{X}^0) \)
10: \( \mathbf{J}^0 \leftarrow J(\mathbf{X}^0) \)
11: \( i_{NR} \leftarrow 1 \)
12: repeat
13: \( \mathbf{X}_{full}^{i_{NR}} \leftarrow \text{SOLVE} \left( \mathbf{J}_{NR}^{i_{NR}} \left( \mathbf{X}_{full}^{i_{NR}} - \mathbf{X}_{NR}^{i_{NR}-1} \right) = -\mathbf{F}_{NR}^{i_{NR}-1} \right) \)
14: \( \mathbf{X}^{i_{NR}} \leftarrow (1 - \lambda_{NR}) \mathbf{X}_{full}^{i_{NR}} + \lambda_{NR} \mathbf{X}_{full}^{i_{NR}} \)
15: \( \mathbf{F}^{i_{NR}} \leftarrow \mathbf{F}(\mathbf{X}^{i_{NR}}) \)
16: \( \mathbf{J}^{i_{NR}} \leftarrow J(\mathbf{X}^{i_{NR}}) \)
17: \( i_{NR} \leftarrow i_{NR} + 1 \)
18: until converged or \( i_{NR} > N_{NR} \)

in which properties are defined. The second reason is particularly important: if the solution is near the domain boundary, an overstep typically causes an immediate failure of the state derivative evaluation, unless appropriate extrapolation procedures are used. This is particularly relevant to the water property evaluation for this benchmark problem: as discussed in section 2.2, the nearly incompressible model only works if \( h_v(T) \) is monotonic. At the operating pressure listed in Appendix A, \( h_v(T) \) stops being monotonic near the saturation temperature. The steady state water temperature of the top axial segment is, by design, close to the saturation temperature, and so a small overstep by a solver iteration takes the state outside of the defined property domain. Therefore, sufficiently small damping factors are important for preventing the solver from stepping outside of the domain.

The initial guess for the steady state solution of the benchmark was taken from the typpwr.dat deck; this guess is not a good one, because of the modifications to the problem listed in the previous section. As discussed above, the steady state of the benchmark problem was found by setting \( d\mathbf{X}/dt \) to zero, replacing the \( dp_b/dt \) term with \( P - P^0 = 0 \), and solving the resulting system with the above solver. With \( N_P = 25 \), \( \lambda_P = 0.25 \) and \( \lambda_{NR} = 1.0 \), the method converged in 25 Picard and 5 Newton-Raphson iterations for full power steady state, and in 25 + 4 iterations for hot zero power steady state. With \( N_{fj} = 10 \), as used in this benchmark, the regular Newton-Raphson method (without Picard iterations) happens to converge without extrapolation required for the water properties, but fails with most other \( N_{fj} \), because one of the iterations takes the method outside of the property domain. For this reason, the use of the above damping factors is recommended for robustness.

These damping factors were chosen through trial and error: generally, the method with a low Krasnosel'ski damping factor will converge slower, but more reliably than a method with the factor closer to 1.0, so choosing a low \( \lambda_P \) allowed the Picard method to come sufficiently close to the solution before being converged by the faster Newton-Raphson. Experience indicates that \( N_P \) between 0 and 50, \( \lambda_P \) between 0.1 and 0.5, and \( \lambda_{NR} = 1.0 \) tend to work best as an initial configuration for the solver, when used on a steady state reactor problems. These parameters often do need to be varied slightly depending on the problem (if it is poorly conditioned and the initial guess is bad, otherwise it will converge regardless), but this is a simple

\[ \text{Algorithm 1 Nonlinear Damped Picard Iteration and Newton-Raphson-based Solver} \]
procedure.

This solver was also used by the time integrator; because following a transient is generally much simpler than searching for a steady state based on a poor initial guess, the Picard iterations and the damping become unnecessary, and the regular Newton-Raphson method is used as the most efficient configuration. If the Newton-Raphson method fails to converge during a time step, it is generally a good indicator that the time step should be shortened. The transient solutions are presented in the next section.

5.3. Transient solutions

The solutions of the transient were used for two different purposes: 1) to study of the convergence of the fully coupled fixed step time integrator, and 2) to compare the results of the simplified model to RELAP5-3D (not typwer.i, but a simplified deck which fully matches the benchmark problem discussed here). A 3rd-order Backward Differentiation Formula (BDF3) time integrator was used to obtain the fixed step transient solution, chosen due to its success in Sosnovsky and Forget (2013). MATLAB’s ode15s adaptive time integrator was also used, to verify the BDF3 solution. Three quantities are of interest: the reactor power $P$, reactivity $\rho$, and the average fuel temperature of the hottest axial segment $T_{f,max}$ (segments 3 and 4 with HZP and VHZP ICs, respectively). Figures 10, 12 and 14 illustrate the observed results for each quantity with HZP ICs, and Figures 11, 13 and 15 give the VHZP ICs solutions.

![Figure 10: Reactivity Transient with Hot Zero Power ICs](image)

Both transients look as expected: 1) reactivity rises linearly with negligible feedback, following $\rho_{ex}(t)$, 2) once the reactor gets prompt critical, power shoots up, which enables rapid thermal feedback, 3) reactivity rapidly drops to subprompt levels, and 4) the reactor stabilizes shortly after the reactivity insertion ends. The BDF3, ode15s and RELAP5-3D (with a fine step) results match rather closely; the disagreements are likely due to the differences in water properties and the HTCs discussed in section 5.1. Additionally, the simplified nature of the thermal hydraulic model may account for some of the discrepancy. This is confirmed by the fact that, at steady state, the disagreements disappear entirely if constant properties are used instead. The fine 1 ms time step used by RELAP5-3D is shown as the time step necessary to converge $T_{f,max}$ to within 10 K at all points in time. The full power steady state $T_{f,4}$ predicted by RELAP5-3D and BGSolver are approximately 15 K apart, which also confirms the difference in thermal hydraulic properties as the reason for the disagreements between RELAP5-3D and BGSolver observed throughout the transients.
A fixed time step of $\Delta t = 5$ ms was used by BDF3; at this point, the solution is clearly converged, as indicated by the comparison with \texttt{ode15s}. A significantly finer maximum time step of $\Delta t_{\text{max}} = 1$ ms is required for RELAP5-3D to converge; as the figures show, a coarser time step resulted in a significant overshoot of the post-peak drop in reactivity, for both transients. This is likely due to the fact that RELAP5-3D is, fundamentally, an operator splitting-based code, while \texttt{BGSolver} is fully coupled. These results
completely repeat the findings of Sosnovsky and Forget (2013): 1) the coarsest time step required by BDF3 here is approximately three quarters of the time step required for the LRA benchmark, in which 3$\frac{3}{4}$ of reactivity was injected in 2 seconds, which is three quarters the speed of this transient, and 2) operator splitting required a nearly an order of magnitude slower time step than a fully coupled time integrator did.

The time step size used by ode15s is shown in Figure 16. The figure clearly illustrates the potential
benefit of an adaptive time integrator with a fully coupled system: while the integrator refines the time step below 5 ms during sharp transitionary periods (when the reactor is superprompt critical and when reactivity injection stops), for a lot of the transient, it uses a significantly coarser time step. This behavior was not observed for RELAP5-3D: the code used the maximum specified time step, despite possessing an adaptive time stepping capability.

The observed convergence of the BDF3 time integrator is studied in greater detail in the next section.

5.4. Convergence results

The convergence of three quantities of interest was studied: peak power, peak reactivity, and final fuel temperature of the hottest axial segment. The convergence of the peaks was analyzed in two ways: the convergence of the quantity of interest at the time of true peak \( t^*_{\text{peak}} \), and the convergence of the peak observed for each time step. To evaluate the quantities at a specific time, and to calculate the observed peaks, cubic interpolation of the computed points was used.

The BDF3 solution with \( \Delta t = 9.765625 \mu s = (5 \text{ ms}) \cdot 0.5^9 \) was used as the “exact” solution. Figures 17, 18 and 19 illustrate the code’s convergence.

The three figures clearly show that BDF3 successfully achieved 3rd-order convergence in time, even for the peak quantities at true peak times, which tend to be the more sensitive parameters. The 3rd-order convergence is present even at the relative coarse time step of 5 ms. The one observed exception is the convergence of \( T_{f4}(t = 3 \text{ s}) \) with VHZP ICs, which only shows linear convergence. This is due to the fact that the VHZP is not a steady state IC, while the BDF3 time integrator initiated with pre-initial states (see Sosnovsky and Forget (2013)) requires an initial steady state to assure superlinear convergence. The peak quantities are much less sensitive to the initial conditions.

The fact that this is a superlinear time integrator is what leads to it nearly converging at the coarse time step, which in the previous section was shown to be too coarse for an operator splitting time integrator (RELAP5-3D).

Excluding the sorting steps, the BDF3 time integration with 5 ms time steps took 24 seconds on an Intel i7-2600 3.4 GHz machine with Windows 7 x64 and MATLAB R2013a x64. The \texttt{ode15s} time integration of the same problem took 2.2 s, and the RELAP5-3D run for the same problem on the same machine with the

Figure 15: Axial Segment 4 Average Fuel Temperature Transient with Hot Zero Power ICs
same maximum time step took 2.1 s. However, as noted in section 5.3 RELAP5-3D is clearly not converged with $\Delta t = 5 \text{ ms}$; with a time step of 1 ms, RELAP5-3D took approximately 12.4 s. This illustrates that while the fixed-step BDF3 integrator clearly loses in performance to the more efficient RELAP5-3D, the two are more comparable (within a factor of two) when a finer, as required, time step is used by RELAP5-3D;
both approaches are clearly beaten by an adaptive time integrator \texttt{ode15s}. It should be noted, that unlike RELAP5-3D, both BDF3 and \texttt{ode15s} are written in MATLAB, and so are expected to greatly accelerate when implemented in a more efficient, fully compiled language.

These results, together with Sosnovsky and Forget (2013), confirm that full coupling through bond graphs
allows for much coarser time steps than would be available through an operator splitting-based approach. 

BGSolver, used in both studies, was successfully used to run these two problems, and so is recommended by the authors as a bond graph processing code for small and medium-sized problems.

6. Conclusions

The bond graph formalism is a technique for modeling engineering systems as combinations of connected elements. After the bond graph representation of a multiphysics system is constructed, it can be automatically converted into a state derivative vector and integrated.

In this paper, we presented a simplified model of a PWR core, which captured the important physics for fast core transients, like rod ejection. The simplified model was successfully compared to a RELAP5-3D model of a similar reactor.

The paper introduced the first bond graph representation of a fully nonlinear neutron point kinetics system, which was used as part of the simplified PWR core model. Together with previously published work, there now exist two types of neutron kinetics bond graph representations: neutron diffusion and point kinetics. The nearly-incompressible thermal hydraulic model presented in this paper will be used as the basis for more advanced coarse thermal hydraulic models to be represented with bond graphs.

To take advantage of the bond graph formalism, a flexible bond graph processing code is required. Such code, BGSolver, was developed and released as open source software; it was first used for previous publications, but is introduced in greater detail in this paper.

A benchmark problem was solved in this paper using the simplified PWR core model and BGSolver. The solution again demonstrated the desired 3rd-order convergence of a fully coupled BDF3 time integrator, unachievable with operator splitting. The solution closely matched a RELAP5-3D reference solution of the same benchmark. Notably, BDF3 was shown to be converged with a 5 ms time step, while RELAP5-3D required a much finer (1 ms) time step to obtain a similar level of convergence. This is consistent with previously observed results for a BWR spatial kinetics problem, also solved with both an operator splitting and a 3rd-order fully coupled time integrator.

Lastly, a simple but robust nonlinear solver algorithm was described, suggested for quick implementations of steady state solvers of fully coupled nonlinear systems. The algorithm was used for both the steady state search and the time integration of the benchmark problem.

BGSolver relies on MATLAB symbolic sorting of the algebraic equations used to construct the state derivative vector. Current work is focused on the possibility of replacing symbolic sorting with a faster and more scalable matrix-based algorithm. Future work will also pursue the implementation of the method in a low-level language to leverage existing high-performance nonlinear solvers.

7. Acknowledgements

The first author received funding from the DOE Office of Nuclear Energy’s Nuclear Energy University Program fellowship.

8. Bibliography


Appendix A. Benchmark problem description

The benchmark problem described here is a modified version of the PWR core from from RELAP5-3D sample decks collection (The RELAP5-3D Code Development Team, 2005, typpwr.i deck distributed with the code).

The problem consists of a PWR core held at fixed pressure, with a constant specified inlet coolant temperature and flow rate. A fixed known spatial power shape is used, justifying the use of a point kinetics neutronic model. Thermal feedback is described by the (simplest) spatially distributed separable linear reactivity feedback model. Only the fuel and moderator temperature feedback is considered; the effects of moderator density are assumed to be fully covered by the moderator temperature feedback, same as in typpwr.i. The physics modeled in the benchmark are described in section 2.

The core geometry is described in Tables A.2 and A.4. The source fractions and feedback coefficients are given in Table A.2, and the precursor family properties in Table A.3. Inlet parameters, including average core pressure, are also given in A.4.

Standard light water properties can be used to describe the coolant; in this study, the IAPWS-IF97 properties (Wagner et al., 2000) were used, which are an update on the IFC-67 properties (International Formulation Committee of the 6th International Conference on the Properties of Steam, 1967) used by RELAP5-3D (by default, including in this paper). The fuel and clad temperature-dependent material properties, taken from typpwr.i, are shown in Figures A.20 and A.21, respectively. A constant thermal conductivity of 1.5258 W m$^{-1}$ K$^{-1}$ and a volumetric heat capacity of 4.3595 J m$^{-3}$ K$^{-1}$ are assumed for the gap.

The reactor’s nominal full power is 3600 MW. The initial conditions and the initiating transient are discussed in section 5.1.

Table A.2: Axial Parameters

<table>
<thead>
<tr>
<th>i</th>
<th>$\Delta z_i$ (m)</th>
<th>$N_{pi}$</th>
<th>$S_i$</th>
<th>$\gamma_i$</th>
<th>$\alpha_{fi}$ (K$^{-1}$)</th>
<th>$\alpha_{wi}$ (K$^{-1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.7639</td>
<td>36 552</td>
<td>0.141211</td>
<td>0.003769</td>
<td>$-5.35743 \times 10^{-6}$</td>
<td>$-9.53550 \times 10^{-5}$</td>
</tr>
<tr>
<td>2</td>
<td>0.6655</td>
<td>36 552</td>
<td>0.204599</td>
<td>0.005461</td>
<td>$-8.73639 \times 10^{-6}$</td>
<td>$-15.5376 \times 10^{-5}$</td>
</tr>
<tr>
<td>3</td>
<td>0.6655</td>
<td>36 552</td>
<td>0.208339</td>
<td>0.005561</td>
<td>$-7.80390 \times 10^{-6}$</td>
<td>$-13.8879 \times 10^{-5}$</td>
</tr>
<tr>
<td>4</td>
<td>0.6655</td>
<td>36 552</td>
<td>0.206780</td>
<td>0.005520</td>
<td>$-7.29846 \times 10^{-6}$</td>
<td>$-12.9870 \times 10^{-5}$</td>
</tr>
<tr>
<td>5</td>
<td>0.6655</td>
<td>36 552</td>
<td>0.174414</td>
<td>0.004656</td>
<td>$-7.80741 \times 10^{-6}$</td>
<td>$-13.8879 \times 10^{-5}$</td>
</tr>
<tr>
<td>6</td>
<td>0.6508</td>
<td>36 552</td>
<td>0.038668</td>
<td>0.001032</td>
<td>$-2.46519 \times 10^{-6}$</td>
<td>$-4.38750 \times 10^{-5}$</td>
</tr>
</tbody>
</table>

Table A.3: Precursor Family Properties

<table>
<thead>
<tr>
<th>m</th>
<th>$\lambda_m$ (s$^{-1}$)</th>
<th>$\beta_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0127</td>
<td>0.0002470</td>
</tr>
<tr>
<td>2</td>
<td>0.1150</td>
<td>0.0012220</td>
</tr>
<tr>
<td>3</td>
<td>1.4000</td>
<td>0.0008320</td>
</tr>
<tr>
<td>4</td>
<td>0.0317</td>
<td>0.0013845</td>
</tr>
<tr>
<td>5</td>
<td>0.3110</td>
<td>0.0026455</td>
</tr>
<tr>
<td>6</td>
<td>3.8700</td>
<td>0.0001690</td>
</tr>
</tbody>
</table>
Table A.4: Other Reactor Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_f$ (mm)</td>
<td>4.267</td>
</tr>
<tr>
<td>$\Delta r_g$ (mm)</td>
<td>0.6096</td>
</tr>
<tr>
<td>$\Delta r_c$ (mm)</td>
<td>0.57</td>
</tr>
<tr>
<td>$A_w$ (m$^2$)</td>
<td>4.9617</td>
</tr>
<tr>
<td>$D_{hw}$ (m)</td>
<td>0.0135</td>
</tr>
<tr>
<td>$P/D$</td>
<td>1.1</td>
</tr>
<tr>
<td>$\Lambda$ (s)</td>
<td>$2.18855 \times 10^{-5}$</td>
</tr>
<tr>
<td>$\dot{m}$ (kg s$^{-1}$)</td>
<td>18.39515</td>
</tr>
<tr>
<td>$p_{core}$ (bar)</td>
<td>156.5478</td>
</tr>
<tr>
<td>$T_{w,0}$ (K)</td>
<td>550</td>
</tr>
</tbody>
</table>

* Coolant hydraulic diameter.
* Pitch-to-diameter ratio.

Figure A.20: Fuel Properties
Figure A.21: Clad Properties